A FREE-RUNNING NUMERICALLY-CONTROLLED OSCILLATOR USING COMPLEX MULTIPLICATION WITH COMPENSATION FOR AMPLITUDE VARIATION DUE TO CUMULATIVE ROUND-OFF ERRORS

The present invention relates to numerically controlled oscillators and, more particularly, to the correction of cumulative approximation or round-off errors within numerically controlled oscillators.

The demodulation of a transmitted signal often requires down-conversion to baseband, which in turn requires mixing with, or multiplication by, a locally generated sinusoidal signal having the same carrier or intermediate frequency as the signal to be down-converted. In a digital channel demodulation system, this function is typically performed using a numerically controlled oscillator (NCO). The capability of an NCO is desirable to digitally generate trigonometric functions of a linearly varying phase angle. Exceptions are cases wherein the desired frequency is either a small integer multiple of the digital sampling frequency such that the Nyquist frequency is reached, or a fraction whose numerator and denominator are both small integers.

Digital channel demodulation often requires a free-running source of sinusoidal signals of a given frequency, such as provided by a numerically controlled oscillator. Such an oscillator can be implemented using the successive multiplication of a complex number by a complex constant that corresponds to the desired frequency. A well-known problem with this method is that the amplitude of the resulting signal is unstable, and will either increase or decrease as a result of cumulative approximation or round-off errors. A numerically controlled oscillator (NCO) synthesizes a range of frequencies from a fixed time base.

Numerically controlled oscillators have conventionally been implemented either a look-up table or the CORDIC (Coordinate Rotation Digital Computer) algorithm for complex phasor rotation. Problems exist with either implementation. For a high degree of numerical precision, look-up tables can require large amounts of memory and CORDIC algorithms can require numerous stages of digital hardware. One well-known method of implementing an NCO is

to use a look-up table having a digital memory addressed by the current phase angle of the oscillator, and the contents of the digital memory are the numerical values of the desired sinusoidal function corresponding to each phase angle. The size of the memory increases exponentially with the required phase angle precision (i.e., it doubles in size for every additional binary digit), and linearly with the precision of the sinusoidal value. If arbitrary values are required for the phase angle, the size of the required look-up table can become very large, requiring a large area if implemented on an integrated circuit device. A second well-known method of digitally generating sinusoidal functions is an iterative procedure that uses the CORDIC algorithm, in which the desired sinusoidal function is successively approximated. In conventional implementations, the CORDIC algorithm increases the precision by a factor of two every iteration that is performed. The CORDIC algorithm can be implemented in software executed on a computer. The CORDIC algorithm can also be implemented in a digital circuit, with one stage required per iteration of the CORDIC algorithm performed. The software implementation requires resources in the form of computational overhead such as processor execution time, and the digital circuit implementation requires resources in the form of area on an integrated circuit device. If a high degree of numerical precision is desired, than a substantial amount of resources are required to implement the CORDIC algorithm.

From the foregoing discussion, it should be readily apparent that there remains a need within the art for a numerically controlled oscillator that requires fewer resources than those currently available.

This invention addresses the shortcomings within the prior art where the amplitude of the resulting signal becomes unstable and eventually increases or decreases as a result of cumulative approximation or round-off errors by providing a simple and efficient method of correcting for such variations, so that the oscillator signal amplitude remains constant. The invention employs successive complex multiplications to implement a numerically controlled oscillator that requires a smaller amount of resources than either the use of look-up tables or CORDIC algorithms resulting in a simple and efficient method of

solving the stability problem. The invention can be implemented using custom-designed discrete logic in an integrated circuit, a systolic or other reconfigurable processor array in which as few as only one processor may be required for this function.

These and objects of the invention are provided by efficiently generating complex sinusoids of a desired frequency by multiplying a current phasor by a predetermined value once every sampling interval to create a next phasor, identifying if a zero value condition exist within either real or imaginary components of the next phasor and substituting a complementary component of unity amplitude to complex phasor components that exhibit the zero value condition, identifying if a condition of equality exits between for real and imaginary components of the next phasor and substituting an equal component for both real and imaginary components of the next phasor if the condition of equality is identified, and if the zero value condition of the condition of equality are not found, determining an error factor for real and imaginary components of the next phasor and correcting the real and imaginary components by removing the error factor.

FIG. 1a is a flow diagram illustrating the functions performed by the invention;

FIG. 1b is a flow diagram illustrating the functions performed by the invention;

FIG. 1c is a flow diagram illustrating the functions performed by the invention;

FIG. 2 is a phasor diagram illustrating concepts of the invention;

FIG. 3. is a block diagram for a channel demodulator that can employ the flow diagram of FIG.1a, FIG. 1b and FIG. 1c.

It has been discovered that sinusoidal functions can be digitally generated using fewer resources for the rotation of a complex phasor. In the complex domain, the rotation of a complex phasor by a given angle corresponds to a complex multiplication by a second complex phasor whose value, having real and imaginary components, is determined strictly by the phase angle increment. A free-running oscillator in the discrete-time digital domain corresponds to a repetitive increase in the phase angle by a constant amount during each sampling interval. Such an oscillator can be created by successive multiplications of an

initial complex phasor by the aforementioned second phasor. In this arrangement, the norm, or magnitude, of the resulting phasor must always equal unity. Any variation from unity will result in either a progressive decrease in magnitude, which will eventually become infinitesimal, or a progressive increase that will continue without bounds. It is well known in the arithmetic of complex numbers, that if the magnitude of the second phasor *exactly* equals unity, the resulting magnitude will remain constant indefinitely. However, this constant result is a condition that is only be guaranteed in an abstract mathematical sense, which an actual digital signal processing system cannot realistically achieve. Consequently, the generation of sinusoidal functions within digital systems require a mechanism for preventing, or correcting, the aforementioned long-term increase or decrease in phasor magnitude.

Referring to FIG. 3, a block diagram of a digital channel demodulator 49 has a digital front end 50. The present invention is useful in the area of baseband conversion 55 within front end 50. These and conversion 55 receives a digital input which is also received by the automatic gain control (AGC) 52. AGC 52 provides an analog output. The baseband conversion 55 generally provides without conversion of higher frequencies, typically by mixing signals. Sample rate conversion 53 provides adjustments to frequencies employed by baseband conversion 55. Baseband conversion 55 as shown in Fig. 3 includes a function for carrier recovery to lock in on the down converted frequency. Sample rate conversion 53 will adjust the clock rate to make the same as the sampling rate. Ideally, it is desired that the clock rate and the sampling rate should be the same. The down conversion of a signal to base band results in the entire process being able to proceed at a slower sampling rate. Another function of sample rate conversion 53 as simply to ensure that the clock rate and the sampling rate are the same. Filter 59 will provide a Nyquist filter and decimation of the sample rate converted signal. The Nyquist filter prevents aliasing. For example if you have a 20 MHz sampling rate and a signal having a frequency of less than 5 MHz, then many alternate samples can be removed, however, is necessary to filter the signal first to remove anything above 5 MHz. Timing recovery 57 provides a feedback

loop from filter 59 to sample rate conversion 53. Once signals that have been baseband converted 55 and sample rate conversion 53, the filtered signal will be received by the channel estimation and correction 60. The channel estimation and correction 60 will typically provide channel estimation, time interpolation, and frequency interpolation. Channel decoder 62 will receive the estimated and corrected channel data from channel estimation correction 60. Channel decoder 62 will typically provide an output interface for MPEG data that has been the scrambled and decoded by channel decoder 62. System controller 64 typically provides control signals to front end 50, channel estimation and correction 60 and channel decoder 62. The digital channel demodulator 49 illustrated in FIG. 3 will demodulate a transmitted signal by down converting the signal to a baseband level by baseband conversion 55, which requires mixing with, or multiplication by, a locally generated sinusoidal signal having the same carrier or intermediate frequency as the signal to be down-converted. In conventional digital channel demodulation systems, this function is typically performed using a numerically controlled oscillator (NCO). The capability of an NCO is desirable to digitally generate trigonometric functions of a linearly varying phase angle. The baseband conversion 55 will provide for the mixing of sinusoidal signals in a manner that avoids problems within the prior art methods for conversion of signals to baseband that implemented the NCO as a lookup table or CORDIC implementation. The baseband conversion 55 employed by the preferred by the invention is described below.

Referring to FIG. 1a, the present invention provides a simplified manner of detecting and correcting for the aforementioned long-term variations. The present invention employs known properties of complex numbers to perform the required compensations. The complex product of two complex numbers $x + jy$ is given by the relationship $a + jb$ and $c + jd$ as illustrated in Equations 1a and 1b below.

Equation 1a $x + jy = (a + jb)(c + jd)$

Equation 1b $x + jy = (ac - bd) + j(ad + bc)$

For an initial complex value c + jd successively multiplied by a second complex value a+jb n times, the result is given by Equation 2.

Equation 2 $x + jy = (a + jb)^n(c + jd)$

When this same multiplication is performed n+1 times, the result is given by the relationships shown in Equations 3a and 3b.

Equation 3a $x' + jy' = (a + jb)^{n+1}(c + jd)$

Equation 3b $x'+ jy' = (a + jb)(x + jy)$

A consequence of Equation 3b is that the desired result can be obtained by repeatedly multiplying the current phasor value by a+jb once for every sampling interval. The algorithm 10 shown in FIG. 1 is run every sampling period by the preferred embodiment. If the magnitude of a+jb is exactly unity, the result will be equivalent to a free-running numerically controlled oscillator. Since it is very well known that any complex number can be expressed in the polar form, Equation 4 below corresponds to a phase angle having unity magnitude.

Equation 4 $a + jb = \cos\theta + j\sin\theta = e^{j\theta}$

The value of this complex number for a given phase angle is determined as a function of the ratio of the desired oscillator frequency to the digital sampling rate. The numerically controlled oscillator can be implemented by repeatedly multiplying the complex phasor with this predetermined complex number during every sampling interval. This process requires one complex multiplication, which in turn comprises four real multiplications and two real additions, as demonstrated in the above expression for a complex product. Using current technology, the foregoing process requires fewer resources than either a look-up table or a CORDIC implementation, but still requires that the magnitude of the complex phasor remain at unity.

Satisfaction of the unity magnitude criterion, typically, requires determination of the complex magnitude of the phasor (two multiplications, an addition, and a square root operation), followed by multiplication of the phasor by the reciprocal of this magnitude (two division operations). The resources required for these operations would be prohibitively expensive.

Referring to FIG. 1a, the desired determinations and corrections are performed using well-known properties of complex numbers under appropriate conditions that will subsequently be described. The algorithm 10 of FIG. 1a will begin 12 at predetermined periods which are determioned during the system design process. During the system design process these parameters are determined by identifying how quickly the accumulated round-off errors that require correction. Multiply Current Phasor by a predetermined value 14 will perform the complex operation of multiplying the current phasor value by a+jb once for every sampling interval as shown in the relationship of Equation 3b. Zero value condition 16 will determine if either *real* component is zero or if the *imaginary* component is zero. If the phase angle is an *even* integer multiple of 90 degrees (0 or 180 degrees), the complex phasor is a pure *real* number, and its *imaginary* component is zero. Consequently, the real component *must* equal unity under these conditions. It is a simple matter to detect the condition of a zero for the imaginary component (or of any zero numerical value, for which all binary digits, in the representation used for a digital signal processing system, equal logic 0). Whenever this condition is detected, substitute component of unity amplitude 18 will replace the value of the real component with a value for positive or negative unity, depending upon the *sign* of the original value. If the phase angle is an odd integer multiple of 90 degrees (90 or 270 degrees), zero value condition 16 will detect that the complex phasor is a pure *imaginary* number, and its *real* component is zero. Using the arguments given above, the imaginary component must equal unity (once again, positive or negative), and thus substitute component of unity amplitude 18 will replace the value of the imaginary component with a value for positive or negative unity, depending upon the *sign* of the original value. The resources required for this procedure consist simply of two comparisons with zero to make the determination, and one substitution of a complex number component with a constant value (i.e., unity). Clearly, this is much more reasonable for a practical implementation than the direct methods of the prior art.

A variation of the above discussed implementation results in cases wherein the phase angle of the complex phasor is an odd multiple of 45 degrees (the even multiples of 45 degrees are just the multiples of 90 degrees that were considered above). If complex phasor is an odd multiple of 45 degrees, the absolute value of the real component *must* equal that of the imaginary component and this condition of equality 20 will be detected by the algorithm 10. In particular, the absolute value of the real component and that of the imaginary must *both* have an absolute value of 0.70710678 (the square root of 1/2), this follows from a well-known property of complex numbers. Substituting equal components 24 insures that once the condition of equality is detected, each of the real and imaginary components are be replaced by the positive or negative of 0.70710678, depending upon the sign of the original component. Once again, this is done without regard to the actual values. The resources required for this procedure are simply one equality comparison to make the determination, and two substitutions of complex number components with a constant value. Again, this is a much more reasonable approach for a practical implementation than the direct method of the prior art.

Referring now to FIG. 1b and FIG. 1c, the various steps that are performed by the flowchart are performed in two parts. Each of FIG. 1b and FIG. 1b represent the most preferred embodiments of the invention with the flowchart of FIG. 1b being the most preferred embodiment. In FIG. 1b, the algorithm, generally referred to as 35, will perform multiply current phasor by predetermined value 34 as previously described followed by testing for zero value condition 36, which is the identical test performed in FIG. 1a as previously described. Substituting components of unity amplitude 38 will be performed if the zero condition is found to exist and the condition of equality exit 39 will be performed if the zero condition is false as described in FIG. 1a. The algorithm performed in FIG. 1b is the most preferred because it is the most efficient and requires the fewest resources. In most applications, it is envisioned that the algorithm 35 shown in FIG. 1b will be sufficient because the round-off errors and truncation errors do not have to be constantly fixed, periodic fixes are sufficient.

In FIG. 1c, the steps that were performed in FIG. 1a but not performed in FIG. 1b are performed. The reasoning here is that in certain applications or systems, relying on the 45 degree adjustments alone as shown in FIG. 1b may not trigger a sufficient number of corrections. The steps shown in FIG. 1c will provide corrections at predetermined interals. Iterations of the algorithm 40 shown in FIG. 1c will be performed every so often as determined necessary with begin 42 and perform multiply current phasor by predetermined value 44 in the same manner as described in the foregoing discussion to FIG. 1a. Then determine error factor 46 will calculate the error factor as previously described in the discussion to FIG. 1a and remove error factor will eliminate the calculated error factor. The predetermined interval will be determined during the system design process and the algorithm of FIG. 1c will performed in accoradance with the those parameters. The algorithm of FIG. 1c requires more resources than that of FIG. 1b but will insure that corrections are constantly being made. The algorithm 10 of FIG. 1a provides the combination of both FIG. 1b and FIG. 1c and is also envisioned by the invention.

The substitutions described above can be combined in a single system, resulting in eight possible conditions under which the correction of the invention could be performed instead of just four. The detection points, according to the above schemes, are illustrated in FIG. 2. In Fig. 2, y=0 is illustrated at 0 and 180 degrees. Other zero points are illustrated for x=0 at 90 degrees and 270 degrees. All the foregoing zero points will have unity components substituted within the phasor. At 0 degrees, a unity value of x=1, y=0; at 90 degrees unity value of x=0, y=1; at 180 degrees unity value of x=-1, y= 0; and at 270 degrees unity value of x=0, y=-1; will be substituted for the phasor components. Also shown in Fig. 2, integer multiples of 45 degrees for the phase angle will result in components having equal absolute value being substituted within the phasor. At 45 degrees, a equal absolute value components of x=0.707, y=0.707; at 135 equal absolute value components of x=-0.707, y=0.707; at 225 equal absolute value components of x=-0.707, y=-0.707; and at 315 degrees equal absolute value components of x=0.707, y=-0.707; will be substituted for the phasor components.

For the above-described schemes to work properly, it is necessary that the conditions being detected arise with some regularity; otherwise, the required corrections will never be performed. In order to best insure this, it is necessary that the desired oscillator frequency *NOT* be too closely related to the digital sampling rate of the system. If their ratio is a simple fraction consisting of small integers, it is possible, or even likely, that the complex phase angle will only assume a restricted number of values during operation, and the above detection conditions might not be met. In these cases, either a small look-up table, or discrete digital logic, can be used to implement the numerically-controlled oscillator, so that there is not as great a need for more efficient use of resources.

In order to increase the frequency at which the required conditions are detected, it is possible to examine only the higher-order binary digits, or bits, of the complex phasor components, for which these conditions will arise more often. The number of bits thus examined can be chosen on the basis of the maximum acceptable "jump" in the amplitude of the numerically-controlled oscillator that would be permitted during normal operation. The zero detection is performed by ascertaining that *all* of the selected higher-order bits equal either logic 0 or logic 1 (the latter case corresponds to a very small negative value); these comparisons require a single NOR function and a single AND function, respectively. The equality comparison is performed by ascertaining that *all* of the corresponding high-order bits of the two complex components (or the bitwise inverse of one component, in cases where the two have opposite signs) are equal; this requires two exclusive -OR functions for each considered bit (one for the sign inversion of one component, when required, and one for the actual comparison), and one AND function for the results of all bit comparisons.

The rate of amplitude variation will be fairly slow for any free-numerically-controlled oscillator in which the components of the complex phasor are represented by a reasonable number of binary digits. Accordingly, it is possible to correct the phasor magnitude on a regular basis without need for the detection of special conditions, the arising of which must be waited for. Although the proposed method to be described requires greater resources than the methods

described above, it nevertheless requires far fewer resources than the direct method, and is therefore much more practical.

If the magnitude of the complex phasor changes only a small amount over a period of time as a result of accumulated round-off error, this change can be approximated, and the approximation can be used to perform the required compensation. Equation 5 below gives the relationship for a phasor magnitude of *r,* with real and imaginary components *x and y,* that is perturbed by an increment, ε, *much* smaller than unity.

Equation 5 $r = 1.0 + \varepsilon; \quad \varepsilon << 1.0$

Therefore, the square of the phasor magnitude for *r* is represent by Equation 6.

Equation 6 $r^2 = x^2 + y^2 \approx 1.0$

Accordingly, the increment of the squared modulus is given by the relationships of Equations 7a, 7b and 7c.

Equation 7a $\Delta(r^2) = r^2 - 1.0 = (1.0 + \varepsilon)^2 - 1.0$

Equation 7b $\Delta(r^2) = 1.0r^2 + 2\varepsilon + \varepsilon^2 - 1.0$

Equation 7c $\Delta(r^2) \approx 2\varepsilon$

Therefore, the increment, ε, is the error determination which can be represented by the relationship shown in Equations 8a and 8b. Determine error factor 26 within the preferred embodiment will determine this error factor using the relationships shown in Equations 8a and 8b.

Equation 8a $$\varepsilon \approx \frac{\Delta(r)^2}{2}$$

Equation 8b $$\varepsilon = \frac{\left(x^2 + y^2 - 1.0\right)}{2}$$

The correction factor for the x and y components of the complex phasor can then be represented by the relationship of Equation 9, which is performed by remove error factor 28.

Equation 9 $$\frac{1.0}{r} = \frac{1.0}{1+\varepsilon} \approx 1.0 - \varepsilon$$

The error determination can be performed using two multiplications, two additions, and one right shift operation (division by 2).

Consequently, the corrected complex components x' and y' are given by Equations 10a and 10b.

Equation 10a $x' \approx x(1.0 - \varepsilon) = x - \varepsilon x$

Equation 10: $y' \approx y(1.0 - \varepsilon) = y - \varepsilon y$

The correction can be performed using one multiplication and one subtraction. Clearly, this is much more efficient than the direct method, which requires the extremely expensive square root and reciprocal operations.

This invention is especially useful in the front-end portion of digital signal processing systems that perform channel demodulation in which down-conversion of an input signal by an arbitrary frequency not closely related to the sampling rate is required. It can be implemented in hardware using discrete logic on an integrated circuit, or in software that is executed on a computer processor. In particular, it lends itself to a very efficient implementation on a systolic array consisting of a multiplicity of small processors running at an instruction rate that is not a very large multiple of the sampling rate. On such an array, it can be implemented using one or at most a small number of such processors.